

(19)日本国特許庁(J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号

特開平6-12395

(43)公開日 平成6年(1994)1月21日

(51)Int.Cl.⁵

G 0 6 F 15/16

識別記号

4 3 0

3 2 0 K

庁内整理番号

9190-5L

8840-5L

F I

技術表示箇所

審査請求 未請求 請求項の数7(全 7 頁)

(21)出願番号

特願平4-170610

(22)出願日

平成4年(1992)6月29日

(71)出願人 000001007

キヤノン株式会社

東京都大田区下丸子3丁目30番2号

(72)発明者 榎田 幸

東京都大田区下丸子3丁目30番2号 キヤ

ノン株式会社内

(72)発明者 鈴木 茂夫

東京都大田区下丸子3丁目30番2号 キヤ

ノン株式会社内

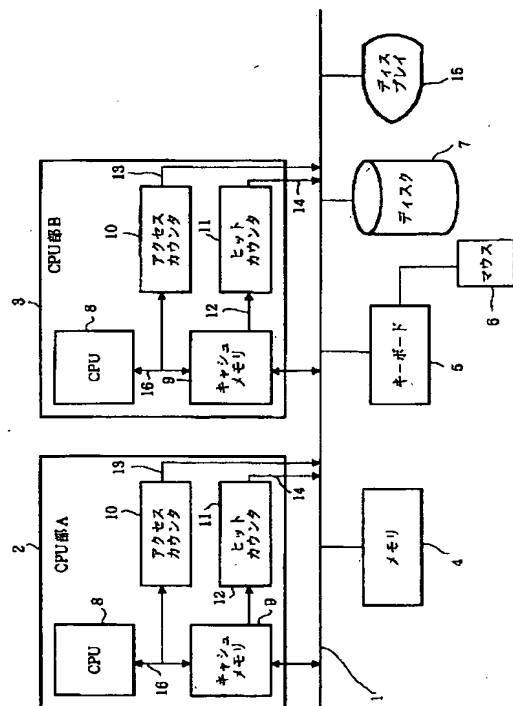
(74)代理人 弁理士 大塚 康徳 (外1名)

(54)【発明の名称】 マルチプロセサシステムにおけるタスク割り付け方法

(57)【要約】

【目的】 マルチプロセサシステムにおいて、発生したタスクをプロセサに割り付ける際にプロセサによる処理効率が良い様に割り付けるプロセサを決定する。

【構成】 CPU 8が処理を行っている際、キャッシュメモリ9にアクセスすると、アクセスカウンタ10によりアクセスの発生した回数が数えられる。また、発生したアクセスのうちで、所望の情報がキャッシュメモリ9内に存在していた回数をヒットカウンタ11で数える。アクセスカウンタ及びヒットカウンタの値から、キャッシュメモリ9に所望の情報があった割合を求める。以上の手順をシステムを構成するCPUすべてについて行い、前記割合の最も低いCPUに新たに発生したタスクを割り付ける。



【特許請求の範囲】

【請求項1】 システムを構成するプロセサ各々について対となるキャッシュ記憶を備えたマルチプロセサシステムにおいて発生したタスクを前記プロセサに割りつけるタスク割り付け方法であって、

前記各プロセサが、前記対となるキャッシュ記憶へアクセスした回数を数える第1の計数行程と、

前記アクセスにより所望される情報が、前記対となるキャッシュ記憶に存在している回数を数える第2の計数行程と、

前記第1の計数工程による回数と前記第2の計数行程による回数とを基に、所望の情報がキャッシュ記憶に存在しているヒット率を算出する算出行程と、

前記ヒット率を基に前記発生したタスクを実行するプロセサを選択する選択行程と、

を備えることを特徴とするタスク割り付け方法。

【請求項2】 前記算出行程はタスクごとに前記ヒット率を算出することを特徴とする請求項1項記載のタスク割り付け方法。

【請求項3】 前記選択行程は、前記各プロセサにおいて前記ヒット率の最も高いタスクのうち、前記ヒット率が最も低いタスクを実行するプロセサを選択することを特徴とする請求項2項記載のタスク割り付け方法。

【請求項4】 前記算出行程はプロセサごとに前記ヒット率を算出することを特徴とする請求項1項記載のタスク割り付け方法。

【請求項5】 前記選択行程は前記ヒット率が最も低いプロセサを選択することを特徴とする請求項4項記載のタスク割り付け方法。

【請求項6】 前記ヒット率の算出はタスクの開始から終了までの時間内のアクセスを対象としてなされることを特徴とする請求項1項記載のタスク割り付け方法。

【請求項7】 前記ヒット率の算出は適当に選ばれた一定の時間内のアクセスを対象としてなされることを特徴とする請求項1項記載のタスク割り付け方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明は、例えばCPUの動作状態を考慮してタスクの割り付けを行なうマルチプロセサシステムに関するものである。

【0002】

【従来の技術】 従来、複数のCPUを有するコンピュータシステム（以下マルチプロセサシステムと呼ぶ）では、1つのマスタとなるCPUとそれに従属するスレーブのCPUとから構成されている場合には、各CPUの処理内容が決められていた。また、各CPUが同格の場合には、空いているCPUがあれば新たに実行すべきタスクをそこに割り付けていた。このとき、もし全CPUが処理中であれば、新規に処理すべきタスクの優先順位と実行中のタスクの優先順位との比較結果から実行タ

クを切り替えたり、比較したタスクの優先順位が同じ場合には、強制的にいずれかのCPUに割り当て、タイムシェア方式により複数のタスクを切り替えながら実行する様に構成されていた。

【0003】

【発明が解決しようとする課題】 しかしながら、上記従来例のマスタスレーブ構成のマルチプロセサシステムでは、あるタスクを実行する場合に、処理内容によるCPUの割り当てのために、空いているCPUがあったとしても、必ず特定のCPUで実行せねばならないことがある。例えばインタラプト処理のようなタスクの場合には、スレーブCPUが空いていても必ずマスタCPUで処理せねばならず、マスタCPUの負荷が増して、システム全体のスループットが落ちると言う欠点があった。

【0004】 一方、各CPUが同格なマルチプロセサシステムにおいて、全CPUが処理を行なっている時に別のタスクの実行を新たに要求された場合には、実行中のタスクの優先順位と新規のタスクの優先順位とが同じであると、例えば各CPUについているデバイス番号の順序といった所定の規則に従ってCPUを割り当てる。新規のタスクは割り当てられたCPU上で既に実行しているタスクとタイムシェアして実行される。このため、CPUの動作状態に関係なく割り当ててしまい、タイムシェア方式によるタスク制御を行なうので、新規タスクのCPUへの割り付けは処理効率についての考慮がなく、非効率的な割り当てがなされる得るという欠点があった。

【0005】 本発明は上記従来例に鑑みてなされたもので、CPUの動作状態を参照して、CPUを効率よく使用できる様にタスクの割り付けを決定できるタスク割り付け方法を提供することを目的とする。

【0006】

【課題を解決するための手段】 上記目的を達成するために、本発明のマルチプロセサシステムにおけるタスク割り付け方法は次のような構成からなる。

【0007】 システムを構成するプロセサ各々について対となるキャッシュ記憶を備えたマルチプロセサシステムにおいて発生したタスクを前記プロセサに割りつけるタスク割り付け方法であって、前記各プロセサが、前記対となるキャッシュ記憶へアクセスした回数を数える第1の計数行程と、前記アクセスにより所望される情報が、前記対となるキャッシュ記憶に存在している回数を数える第2の計数行程と、前記第1の計数工程による回数と前記第2の計数行程による回数とを基に、所望の情報がキャッシュ記憶に存在しているヒット率を算出する算出行程と、前記ヒット率を基に前記発生したタスクを実行するプロセサを選択する選択行程とを備える。

【0008】

【作用】 上記構成により、マルチプロセサシステムを構成する各プロセサがキャッシュメモリにアクセスした際

3

の、所望の情報がキャッシュメモリ上に存在している率を算出し、新たに発生したタスクを割りつけるプロセッサをその率に応じて決定する。

【0009】

【実施例】

【実施例1】本発明の第1の実施例の構成を図1に示す。本実施例では、説明を簡単にするために、CPU部を2個持ったマルチプロセッサシステムを説明する。

【0010】<構成>図中1はシステム内のCPUやメモリ等の資源を接続するためのシステムバスであり、2はCPU部A、3はCPU部Bである。4は各CPU部から共通にアクセスできるメモリ部、5はコマンド等を入力するためのキーボード、6は同じく入力のためのマウス、7はプログラムやデータを格納するためのディスク、15は表示部である。CPU部AとCPU部Bとは同じ構成をしており、図中、同じ構成要素には同じ番号をつけてある。CPU部Aについてその構成を説明する。

【0011】CPU部Aは、CPU8と、メモリアクセスを高速に行なうためにCPU8のローカルなデータを格納しておくキャッシュメモリ9と、CPU8からのメモリアクセス信号16によってキャッシュメモリへのアクセス回数をカウントするアクセスカウンタ10と、CPU8からのメモリアクセス要求に対してキャッシュメモリ9内に該当データがあったことを意味するヒット信号12をとり、キャッシュ・ヒット回数をカウントするヒットカウンタ11とから構成される。カウンタ10の値とカウンタ11の値とはシステムバス1上にマッピングされ、各CPUからアクセスできる様にされている。キャッシュメモリ9は、CPU8からのメモリアクセス要求に対してデータが存在していればヒット信号12を出し、その値をCPU1に出力する。もし存在していなければキャッシュメモリ9はヒット信号12を出さず、メモリ4またはディスク7から該当するデータをキャッシュメモリ9にローディングし、CPU8に求められたデータを返す。またアクセスカウンタ10とヒットカウンタ11とは、CPUに新しいタスクを割り付けた時に共に0にクリアされ、図示していない各CPU部のタイマ等の手段により、所定時間経過ごとに0クリアされる。

【0012】<タスクの割り付け>以上のような構成のマルチプロセッサシステムにおいて、CPU部AでタスクAが、CPU部BでタスクBが実行されている時に、キーボード5やマウス6からの入力でユーザが要求により、あるいは、図示しない外部からの割込みにより新たにタスクCを実行しなければならない場合を考える。

【0013】図2にタスクAとタスクBの各カウンタの変化例を示す。図2によれば、20は、時刻 t_1 におけるCPU部Aのアクセスカウンタ10が100、ヒットカウンタ11が50であることを示し、ヒット率は50

4

／100＝50％であることが分かる。同様にCPU部Bのヒット率は50／150＝33％である。更に時間が経過し、新たなタスクCの実行要求が発生した時の処理を、図6のフローチャートを参照して説明する。

【0014】時刻 t_2 にタスクCの実行要求が発生すると、まずその時点で遊んでいるCPUの有無をテストし、空いているCPUがあればそのCPUで新たに発生したタスクを実行する。すべてのCPU（この場合CPU部AとCPU部B）で何らかのタスクが処理されているなら、それらの中から最適なCPUを選び出さねばならない。

【0015】まず、CPU部AのCPU8がアクセスカウンタ10とヒットカウンタ11の値を読み（S601・S602）、ヒット率を計算する（S603）。CPU部Aでのキャッシュ・ヒット率は表より150／200＝75％となる。この値はキャッシュメモリ9に格納しておき、次にCPU部Bについてヒット率を計算する（S604-NO）。CPU部Aと同じ要領で計算すると、CPU部2のキャッシュヒット率は270／300＝90％となり、これもキャッシュメモリ9に格納しておく。これですべてのCPUについてヒット率の計算が済んだ（S604-YES）。次に各CPU部のヒット率75％と90％とを比較し、小さいヒット率の方を新規タスクを割り当てるCPUと決定する（S605）。最後に小さいヒット率のCPUすなわちCPU部AにタスクCを割り付ける（S606）。これは、キャッシュヒット率の高い方では、キャッシュメモリへの外部の記憶装置からの読み込みが少なく、より高い効率でCPUが稼働している。一方ヒット率の低い方ではキャッシュメモリへの外部記憶からのロードがより頻繁であるため、複数のタスクをタイムシェアして実行する際のオーバーヘッドがヒット率のより高いものほどには全体の処理効率に影響を与えない。このため、高効率の方を優先して処理を進められるように、ヒット率の低いCPUへと新たなタスクを割り付ける。

【0016】これにより、CPU部AはタスクAとタスクCを各タスクの優先順位によりタイムシェア方式により実行し、CPU部BはそのままタスクBを実行する。

【0017】

【他の実施例】第1の実施例では、各CPU部がタスクを1つずつ実行している場合を説明したが、本実施例では、各CPU部が既に複数のタスクを実行している場合を考えてみる。この場合には、CPU部AでタスクAとタスクBとが、CPU部BでタスクCとタスクDとが実行されているものとし、ここにタスクEの実行要求が発生した場合を説明する。

【0018】【実施例2】第2の実施例では、各CPU部で現在いくつのタスクが実行中であることを意識しないで、各CPU部ごとのキャッシュ・ヒット率に従って、どのCPUにタスクEを割り付けるかを決定する方法を

5

説明する。この場合は、第1の実施例と同様に、図3の表に示す様に各CPU部毎のメモリアクセス回数と、キャッシュ・ヒット回数とを図1に示す構成で記憶していけば良く、またタスクを割りつけるCPU部を決定する方法も第1の実施例に示した方式で決定すれば良い。

【0019】図3ではCPU部AとCPU部Bとについてアクセスカウンタとヒットカウンタとが記録されており、ヒット率は実施例1と同じ手順で導くことができ、タスクの割り付け先はCPU部Aと決まる。

【0020】〔実施例3〕第3の実施例では、各タスク毎のキャッシュ・ヒット率を計算し、それから新しいタスクをどのCPU部に割り付けるかの方法を説明する。図4に本実施例の構成図を示す。図4中で図1と同じ動作をする構成要素には同じ番号をつけてある。30のカウンタ制御部は、各CPU部にあるメモリアクセス回数カウンタであるアクセスカウンタ10と、キャッシュ・ヒットの回数カウンタであるヒットカウンタ11とから出力されるカウント情報を各々信号線33・信号線34から入力し、各CPU部のカウント情報を記憶する。さらに各CPUから、各CPUで実行タスクを切り替えたことを知らせる信号35も入力する。カウンタメモリ32は、各CPUで実行中の全タスクのメモリアクセス回数とキャッシュヒット回数を蓄積するためのメモリである。

【0021】以下、詳細な説明をするが、説明のため、unix等のOSで用いられているタスクを識別するためのユニークな識別子（以下タスクIDと呼ぶ）を用いてタスクを識別する。実行中の4タスクについて、タスクAは100、タスクBは102、タスクCは101、タスクDは103とする。カウンタコントローラ31は、システム起動時にはすべてのCPUについて、各CPUからカウンタ情報が入力された時にはそのCPUについて、アクセスカウンタ10、ヒットカウンタ11を0クリアする。タスクAがCPU1で実行され、所定時間経過すると、CPU1はカウンタコントローラ31に対して自分のCPU番号と実行中のタスクID、カウント情報であるアクセスカウンタ10とヒットカウンタ11の値を送る。タスクB・C・Dについても同様な処理を行う。そうすると、カウンタコントローラ31は、カウンタメモリ32上に図5に示すような、或る時刻におけるカウンタを記録したテーブル情報を作成する。図4のアクセスカウンタ10、ヒットカウンタ11のフィールドの意味は第1の実施例と同じである。ここで図4の状態、タスクEの実行要求が発生したとする。そうすると、カウンタコントローラは各タスク毎にキャッシュ・ヒット率を計算する。すなわち、

タスク100: $60/100=60\%$... CPU部A

タスク102: $140/200=70\%$... CPU部A

6

タスク101: $240/300=80\%$... CPU部B

タスク103: $60/200=30\%$... CPU部B

を求める。ここで、4つのキャッシュ・ヒット率の中で1番高いもの、この場合タスク101の80%を見つけ、効率の良いタスクを先に実行するという観点から、タスク101はCPU部Bで実行しているの、CPU部AにタスクEを割り付ける旨を判断し、タスクEをCPU部Aに割り付ける。これは、実施例1の手順をCPU単位からタスク単位へと変更したものである。従って、図6のフローチャートのステップS605を「ヒット率の高いタスクを実行しているCPUを順に除外して、最後に残ったCPUを選ぶ」とすれば、本実施例の適用できる。

【0022】こうして実現されたタスク割り付け方法では、各CPUが実行中のタスクの実行状態として、CPUとキャッシュ間のキャッシュ・ヒット率をモニタする手順を設けることにより、別タスクの実行要求が発生した時に、各CPUで実行中のタスクの優先順位が同じ場合には、キャッシュ・ヒット率の一番低いCPUに今回のタスクを割り付けることに、キャッシュ・ヒット率の高いタスクはそのまま高速に処理を続けることができ、キャッシュ・ヒット率の低いCPUでタイムシェア方式等により処理することにより、このキャッシュ・ヒット率の低いタスクは、もともとキャッシュメモリにあまりヒットしないメモリ・アクセスをしているためにタスクスイッチによるキャッシュ・ヒット率の低下が少なく、効率良くCPU等の資源を割り付けができる。また、キャッシュ・ヒット率を求めるためのハード規模も小さくすむ。

【0023】これまでの説明では、CPUのタイマ等により所定の単位時間を測定し、その単位時間内のキャッシュ・ヒット率を計算する様に説明してきたが、そのタスクの開始から終了まで通したキャッシュ・ヒット率を求める方式でも良い。例えば、大容量の画像データを順次処理して行くようなタスクの場合には、そのタスク全体としてみればキャッシュ・ヒット率は小さいものと考えられるし、また元々時間のかかる処理であるため、他のタスクとタイムシェアして実行しても構わない。

【0024】さらに、キャッシュ・ヒット率を第1・第2の実施例ではCPU部A2で、第3の実施例ではカウンタ制御部30で求めていたが、これに限るものでもない。

【0025】また、これまでの実施例の説明ではCPU部は2個で説明したが、これに限るものでもなく、更に多くのCPUを持った装置であっても良い。また、各CPU上で実行しているタスク数が同じ場合を説明したが、これに限るものでもなく、空いているCPUがあれば、そのCPUに割り付けたりすることは容易に考えら

10

20

30

40

50

7

れる。また、これまではキャッシュ・ヒット率の高いタスクを優先的に実行する様に説明したが、これに限るものでもなく、キャッシュが余りヒットしないタスクは時間がかかるためにそれを優先して実行するという、これまで説明した実施例とは逆の制御方式も容易に考えられる。

【0026】尚、本発明は、複数の機器から構成されるシステムに適用しても1つの機器から成る装置に適用しても良い。また、本発明は、システム或は装置にプログラムを供給することによって達成される場合にも適用できることはいうまでもない。

【0027】

【発明の効果】以上説明した様に本発明に係るタスク割

8

り付け方法は、CPUの動作状態を参照して、CPUを効率よく使用できる様にタスクの割りつけを決定できる。

【図面の簡単な説明】

【図1】第1の実施例のブロック図である。

【図2】第1の実施例のキャッシュヒット率を算出するための表である。

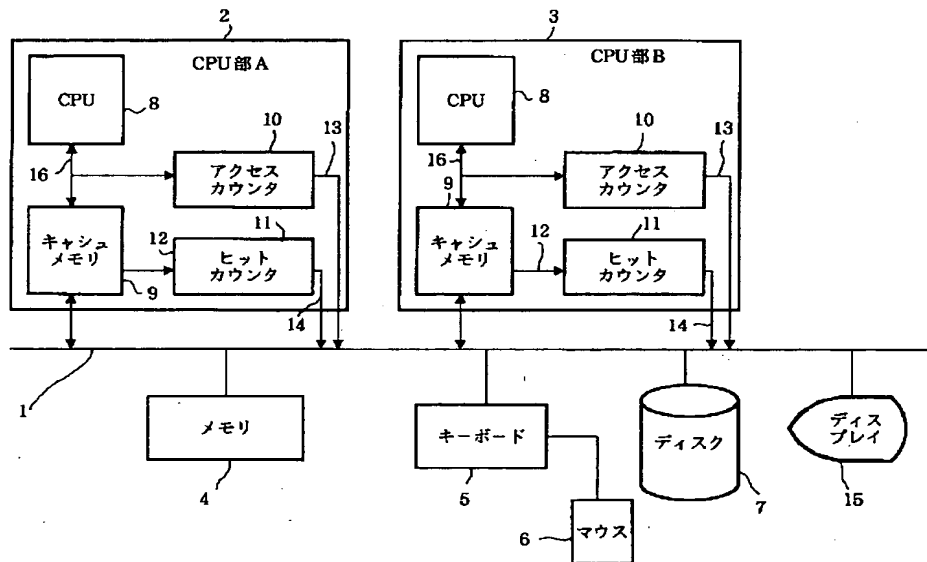
【図3】第2の実施例のキャッシュヒット率を算出するための表である。

【図4】第3の実施例のブロック図である。

【図5】第3の実施例のキャッシュヒット率を算出するための表である。

【図6】第1の実施例のフローチャートである。

【図1】



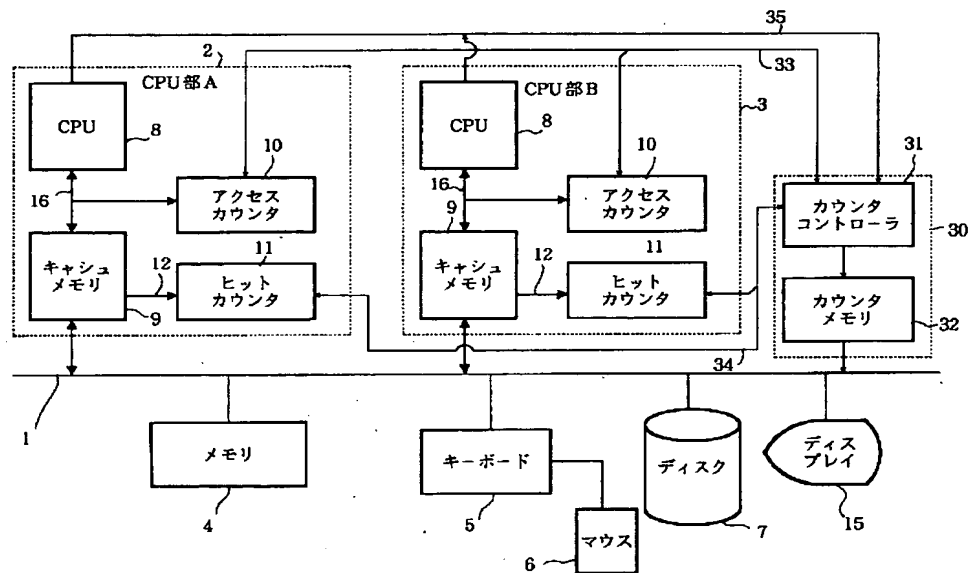
【図2】

時刻 プロセス名 カウナ	プロセスA		プロセスB	
	アクセスカウンタ	ヒットカウンタ	アクセスカウンタ	ヒットカウンタ
t ₁	100	50	150	50
t ₂	200	150	300	270

【図 3】

CPU 名 カウンタ名 時刻	CPU A		CPU B	
	アクセスカウンタ	ヒットカウンタ	アクセスカウンタ	ヒットカウンタ
t_1	300	200	200	900
t_2

【図 4】



【図 5】

CPU 部 A			CPU 部 B		
	アクセス カウンタ	ヒット カウンタ		アクセス カウンタ	ヒット カウンタ
100	100	60	101	300	240
102	200	140	103	200	60

【図6】

